PRINCETON UNIV. F'13 COS 521: ADVANCED ALGORITHM DESIGN

Lecture 2: Karger's Min Cut Algorithm

Lecturer: *Sanjeev Arora* Scribe:*Sanjeev*

Today's topic is simple but gorgeous: Karger's min cut algorithm and its extension. It is a simple randomized algorithm for finding the *minimum cut* in a graph: a subset of vertices S in which the set of edges leaving S, denoted $E(S, \overline{S})$ has minimum size among all subsets. You may have seen an algorithm for this problem in your undergrad class that uses maximum flow. Karger's algorithm is elementary and and a great introduction to randomized algorithms.

The algorithm is this: Pick a random edge, and merge its endpoints into a single "supernode."Repeat until the graph has only two supernodes, which is output as our guess for min-cut. (As you continue, the supernodes may develop parallel edges; these are allowed. Selfloops are ignored.)

Note that if you pick a random edge, it is more likely to come from parts of the graph that contain more edges in the first place. Thus this algorithm looks like a great heuristic to try on all kinds of real-life graphs, where one wants to *cluster* the nodes into "tightly-knit"portions. For example, social networks may cluster into communities; graphs capturing similarity of pixels may cluster to give different portions of the image (sky, grass, road etc.). Thus instead of continuing Karger's algorithm until you have two supernodes left, you could stop it when there are k supernodes and try to understand whether these correspond to a reasonable clustering.

Today we will first see that the above version of the algorithm yields the optimum min cut with probability at least $2/n^2$. Thus we can repeat it say $20n^2$ times, and output the smallest cut seen in any iteration. The probability that the optimum cut is not seen in any repetition is at most $(1 - 2/n^2)^{20n^2} < 0.01$.

Unfortunately, this simple version has running time about n^4 which is not great.

So then we see a better version with a simple tweak that brings the running time down to closer to n^2. The idea is that roughly that *repetition ensures fault tolerance.* The real-life advice of making two backups of your hard drive is related to this: the probability that both fail is much smaller than one does. In case of Karger's algorithm, the overall probability of success is too low. But if run part of the way until the graph has $n/\sqrt{2}$ supernodes, the chance that the mincut hasn't changed is at least $1/2$. So you make two independent runs that go down to $n/\sqrt{2}$ supernodes, and recursively solve both of these. Thus the expected number of instances that will yield the correct mincut is $2 \times \frac{1}{2} = 1$. (Unwrapping the recursion, you see that each instance of size $n/\sqrt{2}$ will generate two instances of size $n/2$, and so on.) Simple induction shows that this 2-wise repetition is enough to bring the probability of success above $1/\log n$.

As you might suspect, this is not the end of the story but improvements beyond this get more hairy. If anybody is interested I can give more pointers.

Also this algorithm forms the basis of other algorithms for other tasks. Again, talk to me for pointers.